PATENT ABSTRACTS OF JAPAN

(11)Publication number : 08-185164
(43)Date of publication of application : 16.07.1996

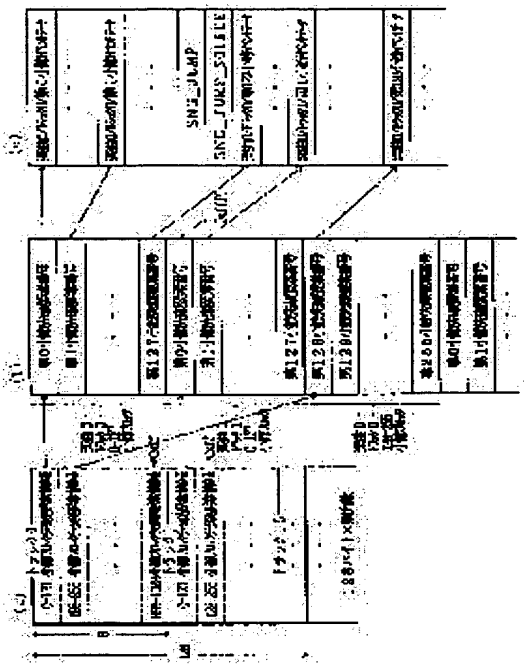
(51)Int.Cl. G10H 1/00

(21)Application number : 06-340418
(22)Date of filing : 29.12.1994

(71)Applicant : CASIO COMPUT CO LTD
(72)Inventor : SETOGUCHI KATSU

(54) AUTOMATIC PLAYING CONTROLLER AND MUSIC DATA STORAGE DEVICE USED FOR THE SAME

(57)Abstract:
PURPOSE: To easily execute an edition operation and to enable the rapid execution of a jump operation of measures, etc., relating to an automatic playing control technique for automatically playing the music data stored into memories, such as sequencers.
CONSTITUTION: The information on the respective top elements of 128 pieces each of arrangement data groups in the arrangement SNG BAR PTR corresponding to respective blocks of the 8 measure blocks obtd. by dividing 1024 measures of the 0-th to the 1023rd measures by 128 measures each is stored in the arrangement SNG BAR TBL by every music and every track within respective pieces of the music. The respective top element numbers of the event data groups within the arrangement SNG SONG corresponding to the respective measures of the 128 measures within the respective measure blocks are stored by every measure block into the arrangement SNG BAR PTR. For example, the increment or decrement of the respective pointer values for the measure blocks on the arrangement SNG BAR TBL or the measures on the arrangement SNG BAR PTR is merely necessitated at the time of fast feed playback or rewinding playback of the measure units.



LEGAL STATUS

[Date of request for examination]	02.07.2001
[Date of sending the examiner's decision of rejection]	
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]	abandonment
[Date of final disposal for application]	20.01.2004
[Patent number]	
[Date of registration]	
[Number of appeal against examiner's decision of rejection]	
[Date of requesting appeal against examiner's decision of rejection]	
[Date of extinction of right]	

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（A）**　(11)特許出願公開番号

特開平8－185164

(43)公開日　平成８年(1996)７月16日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ１０Ｈ　1/00	１０２　Ｚ			

審査請求　未請求　請求項の数４　ＦＤ　（全 20 頁）

(21)出願番号　特願平6－340418

(22)出願日　平成６年(1994)12月29日

(71)出願人　000001443
カシオ計算機株式会社
東京都新宿区西新宿２丁目６番１号

(72)発明者　瀬戸口　克
東京都羽村市栄町３丁目２番１号　カシオ計算機株式会社羽村技術センター内

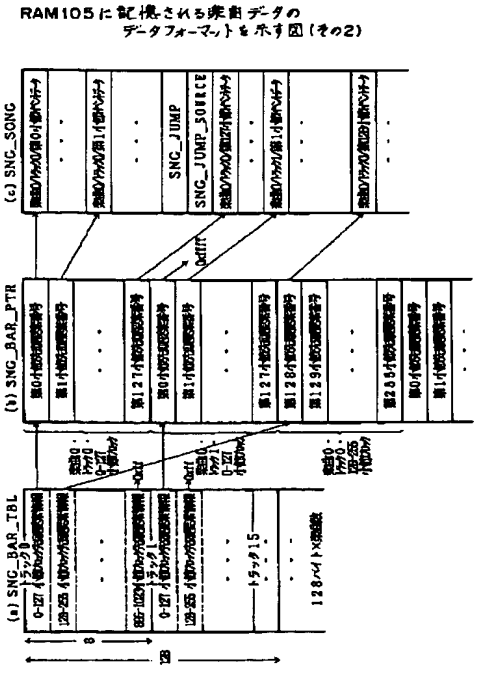
(74)代理人　弁理士　大菅　義之

(54)【発明の名称】　自動演奏制御装置及びそれに使用する楽曲データ記憶装置

(57)【要約】

【目的】　シーケンサー等のメモリに格納された楽曲データを自動演奏するための自動演奏制御技術に関し、編集作業を簡単に実行し、小節単位のジャンプ操作なども迅速に実行可能とすることを目的とする。

【構成】　配列ＳＮＧ＿ＢＡＲ＿ＴＢＬには、楽曲毎及び各楽曲内のトラック毎に、第０小節～第１０２３小節の１０２４小節を１２８小節ずつ分割して得られる８つの小節ブロックのそれぞれに対応する、配列ＳＮＧ＿ＢＡＲ＿ＰＴＲ内の１２８個ずつの配列データ群の各先頭要素情報が記憶される。配列ＳＮＧ＿ＢＡＲ＿ＰＴＲには、小節ブロック毎に、各小節ブロック内の１２８小節のそれぞれに対応する、配列ＳＮＧ＿ＳＯＮＧ内のイベントデータ群の各先頭要素番号が記憶される。例えば小節単位の早送り再生又は巻き戻し再生時には、配列ＳＮＧ＿ＢＡＲ＿ＴＢＬ上の小節ブロック又は配列ＳＮＧ＿ＢＡＲ＿ＰＴＲ上の小節に対する各ポインタ値をインクリメント又はデクリメントするだけでよい。



1

【特許請求の範囲】

【請求項１】　自動演奏を行うための楽曲データに対して自動演奏のための制御を行う自動演奏制御装置において、

小節単位の楽曲データ群であって、それぞれ小節の先頭からの時間データを含むことのできる楽曲データ群を記憶する第１の記憶手段と、

前記小節単位の楽曲データ群の前記第１の記憶手段上での記憶位置を示す情報である小節単位楽曲データ群指示データを記憶する第２の記憶手段と、

該第２の記憶手段に記憶される小節単位楽曲データ群指示データを操作することによって、前記第１の記憶手段に記憶される楽曲データ群に対して自動演奏のための制御を実行する制御手段と、

を有することを特徴とする自動演奏制御装置。

【請求項２】　前記第２の記憶手段に記憶される小節単位楽曲データ群指示データは、連続して記憶される複数個ずつが小節ブロックとして管理され、

該小節ブロック単位の小節単位楽曲データ群指示データの前記第２の記憶手段上での記憶位置を示す情報である指示データを記憶する第３の記憶手段を更に有し、

前記制御手段は、該第３の記憶手段に記憶される指示データ及び前記第２の記憶手段に記憶される小節単位楽曲データ群指示データを操作することによって、前記第１の記憶手段に記憶される楽曲データ群に対して自動演奏のための制御を実行する、

ことを特徴とする請求項１に記載の自動演奏制御装置。

【請求項３】　自動演奏を行うための楽曲データに対して自動演奏のための制御を行う自動演奏制御装置に使用する楽曲データ記憶装置において、

小節単位の楽曲データ群であって、それぞれ小節の先頭からの時間データを含むことのできる楽曲データ群を記憶する第１の記憶手段と、

前記小節単位の楽曲データ群の前記第１の記憶手段上での記憶位置を示す情報である小節単位楽曲データ群指示データを記憶する第２の記憶手段と、

を有することを特徴とする楽曲データ記憶装置。

【請求項４】　前記第２の記憶手段に記憶される小節単位楽曲データ群指示データは、連続して記憶される複数個ずつが小節ブロックとして管理され、

該小節ブロック単位の小節単位楽曲データ群指示データの前記第２の記憶手段上での記憶位置を示す情報である指示データを記憶する第３の記憶手段を更に有する、

ことを特徴とする請求項３に記載の楽曲データ記憶装置。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、シーケンサー等のメモリに格納された楽曲データを自動演奏するための自動演奏制御技術に関する。

2

【０００２】

【従来の技術】従来、楽曲を電子楽器などに自動演奏させるための自動演奏装置においては、楽曲データが以下のような一般的なデータフォーマットでメモリに記憶されている。即ち、例えばスタンダードＭＩＤＩファイルでは、ノートオン、ノートオフなどのイベント間の時間（デルタタイム）が、イベントデータの間に挟まれたデータフォーマットを有する。

【０００３】また、ノートオンからノートオフまでの時間はゲートタイムとして有し、次のノートオン又は他のイベントまでの時間をステップタイムの形式で有するデータフォーマットも知られている。

【０００４】しかし、これらのデータフォーマットは、いずれもイベントデータが単に再生されるのみであれば手軽でよい。

【０００５】

【発明が解決しようとする課題】しかし、イベントデータに対して編集操作を実行したり、ランダムな小節へのジャンプ操作を実行しようとした場合に、上述のデータフォーマットに対してこれらの操作に対応するデータ処理を実行するには、非常に煩雑かつ複雑な制御を要するという問題点を有している。

【０００６】本発明の課題は、編集作業を簡単に実行し、小節単位のジャンプ操作なども迅速に実行可能とすることにある。

【０００７】

【課題を解決するための手段】本発明は、自動演奏を行うための楽曲データに対して、楽曲データの記録、再生、編集などの自動演奏のための制御を行う自動演奏制御装置を前提とする。

【０００８】そして、まず、小節単位の楽曲データ群であって、それぞれ小節の先頭からの時間データを含むことのできる楽曲データ群（イベントデータ群）を記憶する第１の記憶手段（配列ＳＮＧ＿ＳＯＮＧを記憶するＲＡＭ１０５）を有する。

【０００９】次に、小節単位の楽曲データ群の第１の記憶手段上での記憶位置を示す情報である小節単位楽曲データ群指示データを記憶する第２の記憶手段（配列ＳＮＧ＿ＢＡＲ＿ＰＴＲを記憶するＲＡＭ１０５）を有する。

【００１０】そして、その第２の記憶手段に記憶される小節単位楽曲データ群指示データを操作することによって、第１の記憶手段に記憶される楽曲データ群に対して自動演奏のための制御を実行する制御手段（ＣＰＵ１０３等）を有する。

【００１１】本発明は、上述の発明の構成に加えて、更に次のような構成を有することができる。まず、第２の記憶手段に記憶される小節単位楽曲データ群指示データは、連続して記憶される複数個ずつが小節ブロックとして管理される。

【００１２】次に、その小節ブロック単位の小節単位楽曲データ群指示データの第２の記憶手段上での記憶位置を示す情報である指示データを記憶する第３の記憶手段（配列ＳＮＧ＿ＢＡＲ＿ＴＢＬを記憶するＲＡＭ１０５）を更に有する。

【００１３】そして、前述の制御手段は、その第３の記憶手段に記憶される指示データ及び第２の記憶手段に記憶される小節単位楽曲データ群指示データを操作することによって、第１の記憶手段に記憶される楽曲データ群に対して自動演奏のための制御を実行する。

【００１４】なお、上記第１、第２、又は第３の記憶手段のみからなる楽曲データ記憶装置も、本発明の範囲である。

【００１５】

【作用】例えば楽曲データの再生処理において、小節単位の早送り再生又は巻き戻し再生が実行される場合には、制御手段が、第２の記憶手段に記憶される小節単位楽曲データ群指示データ又は第３の記憶手段に記憶される指示データをインクリメント又はデクリメントするだけでよい。この結果、非常に高速な１又は複数小節単位の早送り再生又は巻き戻し再生を簡単な制御で実現することができる。

【００１６】次に、楽曲データの編集処理が実行される場合を考える。例えば小節単位の移動を行うためには、制御手段が、第２の記憶手段に記憶される小節単位楽曲データ群指示データ又は第３の記憶手段に記憶される指示データの並びを入れ替えるだけでよい。この場合、各小節の楽曲データ群に記憶される時間データはそれぞれの小節の先頭からの時間データとすることができるため、各楽曲データの時間データを編集し直す必要もない。

【００１７】更に、例えば既存の小節に新たな楽曲データを追加するためには、制御手段は、第１の記憶手段の末尾に楽曲データ群を追加し、それに対応するデータを第２又は第３の記憶手段に設定すればよい。

【００１８】その他、楽曲データの挿入、削除等も、簡単なデータ操作で実現できる。このように、楽曲データが、第１の記憶手段、第２の記憶手段、又は第３の記憶手段を介して階層的に管理されることにより、楽曲データの再生処理、編集処理等の制御を簡単に実行することができる。

【００１９】

【実施例】以下、図面を参照しながら本発明の実施例につき詳細に説明する。図１は、本発明の実施例の全体構成図である。この実施例は、シーケンサ装置として実現されている。

【００２０】ＣＰＵ１０３は、ＲＯＭ１０４に記憶されている制御プログラムに基づいて、システム全体の動作を制御する。そして、楽曲データの記録時には、特には図示しない電子楽器から出力された演奏情報が、例えば

MIDI(Musical Instrument Digital Interface)規格に基づいて転送され、その演奏情報が演奏情報入力部１０１によって受信される。ＣＰＵ１０３は、この演奏情報を取り込み、その演奏情報を示すイベントデータを、必要に応じてその入力タイミングと共に、ＲＡＭ１０５の楽曲データ記録領域上の記録対象トラックに順次記録する。記録対象トラックは、ユーザが予めシステム操作子群１０７を操作して指定する。

【００２１】楽曲データの編集時には、ＣＰＵ１０３は、ユーザがシステム操作子群１０７において指定した内容に従って、ＲＡＭ１０５の楽曲データ記録領域上の所望のトラックに記録されている楽曲データを読み出して、例えば５線譜の形式で表示部１０２に表示する。ユーザは、システム操作子群１０７を用いて、表示部１０２に表示されている楽曲データに対して、楽曲データの挿入、削除、移動、複写などを行うことができる。この編集結果は、ＲＡＭ１０５の楽曲データ記録領域の楽曲データの内容に反映される。

【００２２】楽曲データの再生時には、ＲＡＭ１０５の楽曲データ記録領域上の再生対象トラックから、イベントデータが順次読み出され、各イベントデータに対応する演奏情報が、例えばMIDI規格に基づいて演奏情報出力部１０６から外部の電子楽器又は音源装置に出力され、これらの装置によってその演奏情報に対応する自動演奏が行われる。

【００２３】図２～図５は、ＲＡＭ１０５に記憶される楽曲データのデータフォーマットを示す図である。本実施例では、楽曲データは、各小節のイベントデータと、連続する１２８小節分のイベントデータを単位とする小節ブロックと、最大で８個の小節ブロックを含むトラック、１６トラックを単位とする楽曲という各単位を用いて、階層的に管理される点が特徴である。

【００２４】まず、図２に示されるトラック使用状況データＳＮＧ＿ＴＲＫ＿ＵＳＥは、１６ビットのデータであって、各楽曲毎に設定される。各データには、トラック０～１５の各トラックが楽曲の自動演奏において使用されるか否かが、“１”又は“０”のフラグによって設定される。このデータの、最下位ビットがトラック０に対応し、最上位ビットがトラック１５に対応する。

【００２５】図３(a)に示される配列ＳＮＧ＿ＢＡＲ＿ＴＢＬには、楽曲毎及び各楽曲内のトラック毎に、第０小節～第１０２３小節の１０２４小節を１２８小節ずつ分割して得られる８つの小節ブロックのそれぞれに対応する、図３(b)に示される配列ＳＮＧ＿ＢＡＲ＿ＰＴＲ内の１２８個ずつの配列データ群の各先頭要素情報が記憶される。配列ＳＮＧ＿ＢＡＲ＿ＰＴＲに定義可能な小節ブロックの数は、例えば最大で３２である。その場合、配列ＳＮＧ＿ＢＡＲ＿ＴＢＬには、小節ブロック先頭要素情報として、０ｘ００～０ｘ１ｆ（１０進数の０～３１に対応）の何れかの値が記憶される。この小節ブ

5

ロック先頭要素情報の値を１２８倍して得られる値が、その小節ブロックに対応する配列ＳＮＧ＿ＢＡＲ＿ＰＴＲ内の１２８個の配列データ群の先頭要素番号となる。但し、未使用のトラックのそれぞれの小節ブロックの先頭要素情報及び使用トラック内の未使用の小節ブロックの先頭要素情報としては、値０ｘｆｆが設定される。

【００２６】図３(b) に示される配列ＳＮＧ＿ＢＡＲ＿ＰＴＲには、上述の小節ブロック毎に、各小節ブロック内の１２８小節のそれぞれに対応する、図３(c) に示される配列ＳＮＧ＿ＳＯＮＧ内のイベントデータ群の各先頭要素番号が記憶される。配列ＳＮＧ＿ＳＯＮＧに定義可能なイベントデータの数は、例えば最大で１６３８４イベントである。その場合、配列ＳＮＧ＿ＢＡＲ＿ＰＴＲには、各小節に対応するイベントデータ群の先頭要素番号として、０ｘ０００００～０ｘ４０００（１０進数の０～１６３８３に対応）の何れかの値が記憶される。但し、全休符の小節に対応する先頭要素番号としては、値０ｘｆｆｆｆが設定される。

【００２７】図３(c) に示される配列ＳＮＧ＿ＳＯＮＧには、上述したように小節単位で連続したイベントデータが記憶される。１つのイベントデータは、後述するように３ワードのデータとして構成される。但し、配列ＳＮＧ＿ＳＯＮＧは、５１２個のイベントデータを単位とする３２個のイベントデータブロック（１６３８４イベントデータ分）に分割される。ここで、１つの小節内のイベントデータ群が２つのイベントデータブロックに分割される場合には、第１番目のイベントデータブロックの末尾に後述するジャンプイベントデータが記憶され、また、第２番目のイベントデータブロックの先頭に後述するジャンプソースイベントデータが記憶される。

【００２８】図４は、図３(c) の配列ＳＮＧ＿ＳＯＮＧに記憶されるイベントデータ群を構成するノートイベントデータとコントロールイベントデータのデータフォーマットを示す図であり、各イベントデータは３ワードのデータとして構成される。

【００２９】図４(a) に示されるノートイベントデータは、楽譜の音符に対応するものであり、単音の発音処理に必要な複数の情報を含む。即ち、ノートイベントデータの第１ワードは、上位７ビットが音高番号、下位７ビットがベロシティを表わす。ノートイベントデータの第２ワードは、そのノートイベントデータに対応する音符について、それが含まれる小節の先頭からの時間位置を、適当な単位（例えば４８０分の１拍のクロック）でカウントした値である。そして、ノートイベントデータの第３ワードは、そのノートイベントデータに対応する音符の発音時間を前述の適当な単位でカウントしたゲートタイムである。

【００３０】図４(b) に示されるコントロールイベントデータは、楽曲の再生を制御するために必要な、上述のノートイベントデータ以外のイベントデータであり、第

6

１ワードの最上位ビットが１であることによって、ノートイベントデータと区別される。このコントロールイベントデータは、更に、メタイベントデータ群、ジャンプイベントデータ群、及びコマンドイベントデータ群の３つに分類される。メタイベントデータ群は、楽曲全体のテンポ、拍子、調に関する情報を含む。ジャンプイベントデータ群は、図５で後述するように、トラック終了、小節終了、次処理データへのジャンプ等の、イベントデータの位置制御に関する情報を含む。更に、コマンドイベントデータ群は、音量、プログラムチェンジ（音色）、ベンダー等の、楽音に対する修飾的な効果を付加するための情報を含む。コントロールイベントデータの第１ワードの下位７ビットはコントロールコードを示し、このコードによって制御内容が決定される。例えば、メタイベントデータ群のテンポイベントデータに対応する第１ワードの値は０ｘｆｆ００、ジャンプイベントデータ群の小節終了イベントデータに対応する第１ワードの値は０ｘｆｆ１１（図５(b) 参照）、コマンドイベントデータ群のプログラムチェンジイベントデータに対応する第１ワードの値は０ｘｆｆ２０となる。コントロールイベントデータの第２ワードは、そのコントロールイベントデータに対応する制御が実行される小節の先頭からの時間位置を、前述した適当な単位でカウントした値である。コントロールイベントデータの第３ワードには、コントロールコード以外に必要なコントロール値が格納される。例えばメタイベントデータ群のテンポイベントデータの第３ワードにはテンポ番号、ジャンプイベントデータ群のジャンプイベントデータの第３ワードには配列ＳＮＧ＿ＳＯＮＧ（図３(c)）内でのジャンプ先のイベントデータの要素番号、コマンドイベントデータ群のプログラムチェンジイベントデータの第３ワードには音色番号が格納される。

【００３１】次に、上述したコントロールイベントデータのうちのジャンプイベントデータ群を構成する４種類のイベントデータについて説明する。図５は、これら４種類のイベントデータのそれぞれのデータフォーマットを示す図である。

【００３２】まず、図５(a) のトラック終了イベントデータは、各トラックのイベントデータ群の終了位置にそのトラックの最終のイベントデータとして挿入される。次に、図５(b) の小節終了イベントデータは、各小節のイベントデータ群の終了位置にその小節の最終イベントデータとして挿入される。

【００３３】更に、図５(c) のジャンプイベントデータと図５(d) のジャンプソースイベントデータは、図３(c) の配列ＳＮＧ＿ＳＯＮＧにおいて、１つの小節内のイベントデータ群が５１２イベントずつの２つのイベントデータブロックに分割される場合に、それぞれ、第１番目のイベントデータブロックの末尾と第２番目のイベントデータブロックの先頭に記憶される。この場合、ジ

7

ャンプイベントデータの第３ワードには配列ＳＮＧ＿ＳＯＮＧ内でのジャンプ先のイベントデータの要素番号が格納され、ジャンプソースイベントデータの第３ワードには同じくジャンプ元のイベントデータの要素番号が格納される。

【００３４】なお、図５(a) ～(d) に示される４種類のイベントデータについて、イベントデータの編集処理及び再生処理においては、第１ワードにより現在処理イベントデータがジャンプイベントデータ群に属するイベントであることが認識された時点で、各々のイベントデータに対応した処理が即座に実行される。従って、これら各イベントデータの第２ワードは任意の値でよい。また、トラック終了イベントデータと小節終了イベントデータの第３ワードも任意の値でよい。

【００３５】以上説明した図２～図５の楽曲データのデータフォーマットが採用されることにより、自動演奏に関する制御を非常に効率的に実行することが可能となる。この効果について、以下に説明する。

【００３６】まず、イベントデータの再生処理時（楽曲の自動演奏時）の動作について説明する。ユーザが図１のシステム操作子群１０７を操作することにより再生開始が指示されると、図１のＣＰＵ１０３は、ＲＡＭ１０５に記憶されている再生される楽曲に対応する図２に示されるトラック使用状況データＳＮＧ＿ＴＲＫ＿ＵＳＥの内容を調べる。ＣＰＵ１０３は、トラック使用状況データＳＮＧ＿ＴＲＫ＿ＵＳＥにおいて値１が設定されているビットに対応するトラックの処理タイミングで、そのトラックのイベントデータを再生させるために、まず、図３(a) に示される配列ＳＮＧ＿ＢＡＲ＿ＴＢＬを探索する。この結果、ＣＰＵ１０３は、現在再生中の小節に対応する図３(b) に示される配列ＳＮＧ＿ＢＡＲ＿ＰＴＲ内のの要素番号を得る。次に、配列ＳＮＧ＿ＢＡＲ＿ＰＴＲ内の該当データの値が０ｘｆｆｆｆでなければ（全休符小節でなければ）、ＣＰＵ１０３は、配列ＳＮＧ＿ＢＡＲ＿ＰＴＲの該当データの値が指す要素番号の配列ＳＮＧ＿ＳＯＮＧ中のイベントデータを読み込む。そして、その第２ワードに格納されている小節先頭からのクロックの値が現在のクロックの値以下であれば、ＣＰＵ１０３は、そのイベントデータを再生するために、そのイベントデータの第１ワードのデータに従った演奏情報を、図１の演奏情報出力部１０６から出力させる。以下、上述の一連の処理が繰り返し実行され、現在小節の現在クロックに従ったイベントデータの再生処理が実行される。

【００３７】上述のイベントデータの再生処理において、小節単位の早送り再生又は巻き戻し再生が実行される場合を考える。この動作を実現するためには、図３(a) に示される配列ＳＮＧ＿ＢＡＲ＿ＴＢＬ上の小節ブロックに対する現在処理ポインタ値、又は図３(b) に示される配列ＳＮＧ＿ＢＡＲ＿ＰＴＲ上の小節に対する現在処理ポインタ値をインクリメント又はデクリメントするだけでよい。この結果、非常に高速な１又は複数小節単位の早送り再生又は巻き戻し再生を簡単な制御で実現することができる。

8

【００３８】次に、イベントデータの編集処理が実行される場合を考える。例えば小節単位の移動を行うためには、図３(a) の配列ＳＮＧ＿ＢＡＲ＿ＴＢＬ又は図３(b) の配列ＳＮＧ＿ＢＡＲ＿ＰＴＲの配列データの要素を入れ替えるのみでよい。この場合、図３(c) の配列ＳＮＧ＿ＳＯＮＧ内のイベントデータに記憶されているクロックデータは小節の先頭からのクロックであるため（図４等参照）、時間データを編集し直す必要もない。更に、例えば既存の小節に新たなイベントデータを追加するためには、図３(c) の配列ＳＮＧ＿ＳＯＮＧの末尾にイベントデータ群を追加し、ジャンプイベントデータとジャンプソースイベントデータ（図５参照）を用いて、そのイベントデータ群を所望の小節のイベントデータ群に連結するだけでよい。その他、イベントデータの挿入、削除等も、簡単なデータ操作で実現できる。

【００３９】このように、楽曲データが階層的に管理されることにより、イベントデータの再生処理、編集処理等の制御を簡単に実行することができる。ここで、上述の楽曲データの階層構造が採用される場合におけるイベントデータの編集処理において、小節内でイベントデータの挿入、削除、移動等の編集作業が実行された場合、図３(c) の配列ＳＮＧ＿ＳＯＮＧ内の１つの小節内のイベントデータ群において、図５に示されるジャンプイベントデータ群以外のイベントデータの第２ワードに設定されている小節の先頭からのクロックの値の大小関係が逆転している状況が発生することがある。例えば、所望の小節内の任意の時間位置に新たなイベントデータが追加される場合、配列ＳＮＧ＿ＳＯＮＧの末尾に、ジャンプソースイベントデータと、それに続いて、第２ワードに所望のクロック値が設定された所望のイベントデータと、更にそれに続いて、小節終了イベントデータが追加されると共に、上記所望の小節の最終イベントの小節終了イベントデータが、第３ワードに上記ジャンプソースイベントデータの要素番号が設定されたジャンプイベントデータに置き換えられ、最後に、上記ジャンプソースイベントデータの第３ワードに上記ジャンプイベントデータの要素番号が設定される。この場合、上記追加イベントデータは、上記所望の小節内のイベントデータ群の最後に単純に追加されただけであるため、その追加イベントデータが上記所望の小節内の他のイベントデータ群との間で正しい時間関係を有するように、その追加イベントデータを含む上記所望の小節内のイベントデータ群が、各イベントデータの第２ワードのクロック値に関して並び換えられる（ソートする）必要がある。

【００４０】この場合に、小節内のイベントデータ群に含まれ得る図５に示されるジャンプイベントデータ群が

適切に処理される必要がある。そこで、本実施例においては、小節内のイベントデータ群の時間ソート処理が次のようにして実行される。以下に、小節内のイベントデータ群の時間ソート処理の詳細を説明する。

【００４１】図６は、図１に示されるＣＰＵ１０３がＲＯＭ１０４に記憶された制御プログラムを実行する動作として実現される時間ソート処理の全体動作フローチャートである。

【００４２】まず、時間ソート処理終了フラグ（ＣＰＵ１０３内のレジスタ又はＲＡＭ１０５に確保される）は、その値が０であるときに時間ソート処理が終了していないことを示し、その値が１であるときに時間ソート処理が終了したことを示す。

【００４３】まず、ステップ６０１において、時間ソート処理終了フラグの値が０にセットされる。次に、ステップ６０２で時間ソート処理終了フラグの値が０であると判定される間、ステップ６０３～６０５の一連の処理が繰り返し実行される。

【００４４】ステップ６０３では、時間ソート処理終了フラグの値が１にセットされる。ステップ６０４では、配列ＳＮＧ＿ＳＯＮＧに記憶されている時間ソート処理の対象とされる小節内のイベントデータ群に含まれ、ジャンプイベントデータ及びジャンプソースイベントデータによって区切られる全てのイベントデータのグループ（以下、イベントデータフラグメント又は単にフラグメントと呼ぶ）のそれぞれの内部で閉じた時間ソート処理であるフラグメント内ソート処理が実行される。

【００４５】続いて、ステップ６０５では、配列ＳＮＧ＿ＳＯＮＧに記憶されている上記対象小節に含まれる隣接するフラグメント同士で、それらの接続部に位置するイベントデータ（ジャンプイベントデータの直前に位置するイベントデータとそのジャンプイベントデータに対応するジャンプソースイベントデータの直後に位置するイベントデータ）についての時間ソート処理であるフラグメント間ソート処理が実行される。

【００４６】そして、上述のステップ６０４のフラグメント内ソート処理とステップ６０５のフラグメント間ソート処理が繰り返し実行され、両方の処理において実際にイベントデータの入替えが発生しなくなった時点で、配列ＳＮＧ＿ＳＯＮＧに記憶されている対象小節内のイベントデータ群に対する時間ソート処理が完了する。具体的には、ステップ６０４又は６０５において実際にイベントデータの入替えが発生した場合は、ステップ６０４又は６０５内において後述するように時間ソート処理終了フラグの値が０にセットされる。従って、この場合にはステップ６０２の判定がＹＥＳとなるため、時間ソート処理が続行される。一方、ステップ６０４及び６０５の両方で実際にイベントデータの入替えが発生しなくなった場合は、ステップ６０４及び６０５の何れにおいても時間ソート処理終了フラグの値を０にセットする処理が実行されなくなる。従って、この場合には時間ソート処理終了フラグの値はステップ６０３で１にセットされたままとなるため、ステップ６０２の判定がＮＯとなり、時間ソート処理が終了する。

【００４７】図７は、図６のステップ６０４のフラグメント内ソート処理の動作フローチャートである。まず、フラグメント内ソート処理終了フラグ（ＣＰＵ１０３内のレジスタ又はＲＡＭ１０５に確保される）は、フラグメント内ソート処理を、終了させるべきでない場合に値０となり、終了させるべきである場合に値１となる。このフラグの値は、ステップ７０１で、０に初期化される。

【００４８】サーチ状態フラグ（ＣＰＵ１０３内のレジスタ又はＲＡＭ１０５に確保される）は、配列ＳＮＧ＿ＳＯＮＧに記憶されている対象小節に含まれる現在処理中のフラグメント内において、ジャンプソースイベントデータ以外の先頭のイベントデータをサーチする処理が、終了していない場合に値０となり、終了している場合に値１となる。このフラグの値は、ステップ７０２で、０に初期化される。

【００４９】サーチイベント（ＣＰＵ１０３内のレジスタ又はＲＡＭ１０５に確保される）は、配列ＳＮＧ＿ＳＯＮＧに記憶されている対象小節に含まれる現在処理中のフラグメント内において、現在処理中のイベントデータを指示するものである。このサーチイベントの値は、ステップ７０３で、配列ＳＮＧ＿ＳＯＮＧに記憶される対象小節内の先頭のイベントデータの要素番号にセットされる。この番号は、図３の配列ＳＮＧ＿ＢＡＲ＿ＴＢＬ及び配列ＳＮＧ＿ＢＡＲ＿ＰＴＲから容易に検索できる。

【００５０】続いて、初期状態では、ステップ７０４と７０５の判定が共にＹＥＳとなった後、ステップ７０６～７１３において、配列ＳＮＧ＿ＳＯＮＧに記憶されている対象小節に含まれる現在処理中のフラグメント内において、ジャンプソースイベントデータ以外の先頭のイベントデータをサーチする処理が実行される。

【００５１】まず、サーチイベントによって指示される現在イベントデータ（フロー中では現在イベントと記述する）が、ＲＡＭ１０５内の配列ＳＮＧ＿ＳＯＮＧから読み出される。

【００５２】そして、現在イベントデータがジャンプソースイベントデータである場合は、ステップ７０６の判定の後にステップ７０７が実行され、サーチイベントが指示する配列ＳＮＧ＿ＳＯＮＧの要素番号がインクリメントされる。これらのステップ７０６とステップ７０７の処理の繰返しによって、現在処理中のフラグメント内の先頭のジャンプソースイベントデータが無視される。

【００５３】現在イベントデータがジャンプソースイベントデータ以外のイベントデータとなった場合は、ステップ７０８で、そのイベントデータの種類が判定され

11

る。ステップ７０８で現在イベントデータがトラック終了イベントデータ又は小節終了イベントデータ（図５参照）であると判定された場合には、ステップ７０９で、フラグメント内ソート処理終了フラグの値が１にセットされる。即ち、この場合は、現在処理中のフラグメント内にはソート処理を実行すべき有効なイベントデータは存在せず、かつ現在処理中のフラグメント以降に対象小節のイベントデータ群が存在しないため、ステップ７０９の処理の後にステップ７０４が実行された時点でその判定がＮＯとなり、フラグメント内ソート処理を終了する。

【００５４】また、ステップ７０８で現在イベントデータがジャンプイベントデータ（図５(c)参照）であると判定された場合には、ステップ７１０で、そのジャンプイベントデータの第３ワードに格納されているジャンプ先要素番号が、サーチイベントの値としてセットされる。即ち、この場合には、現在処理中のフラグメント内にはソート処理を実行すべき有効なイベントデータは存在せず、かつ現在処理中のフラグメント以降に対象小節のイベントデータ群が存在する。このため、ステップ７１０の処理によってサーチイベントに新たなフラグメントの先頭のイベントデータの要素番号が設定された後に、再びステップ７０４と７０５の判定が共にＹＥＳとなった後、ステップ７０６～７１３において、配列ＳＮＧ＿ＳＯＮＧに記憶されている対象小節に含まれる新たなフラグメント内において、ジャンプソースイベントデータ以外の先頭のイベントデータをサーチする処理が再度実行される。

【００５５】更に、ステップ７０８で現在イベントデータがトラック終了イベントデータ、小節終了イベントデータ、及びジャンプイベントデータ以外のイベントデータであると判定された場合には、ステップ７１１で、サーチイベントの値がフラグメント先頭イベントの値としてセットされる。この値が、現在処理中のフラグメント内で最初に見つかったジャンプソースイベントデータ以外の先頭のイベントデータの要素番号となる。続いて、ステップ７１２で、サーチイベントが指示する配列ＳＮＧ＿ＳＯＮＧの要素番号がインクリメントされた後、サーチ処理を終了させるべく、ステップ７１３で、サーチ状態フラグの値が１にセットされる。

【００５６】この結果、ステップ７１３に続いてステップ７０４の判定がＹＥＳとなった後に、ステップ７０５の判定がＮＯとなって、ステップ７１４～７１６が実行される。ステップ７１４～７１６では、配列ＳＮＧ＿ＳＯＮＧに記憶されている対象小節に含まれる現在処理中のフラグメント内で、トラック終了イベントデータ、小節終了イベントデータ、及びジャンプイベントデータ以外の末尾のイベントデータをサーチする処理が実行される。

【００５７】まず、サーチイベントによって指示される

12

現在イベントデータが、ＲＡＭ１０５内の配列ＳＮＧ＿ＳＯＮＧから読み出される。そして、現在イベントデータがトラック終了イベントデータ、小節終了イベントデータ、及びジャンプイベントデータ以外のイベントデータである場合には、ステップ７１４の判定の後にステップ７１５が実行され、サーチイベントが指示する配列ＳＮＧ＿ＳＯＮＧの要素番号がインクリメントされる。これらのステップ７１４とステップ７０７の処理の繰返しによって、現在処理中のフラグメント内のイベントデータが末尾に向かって順次サーチされる。

【００５８】現在イベントデータがトラック終了イベントデータ、小節終了イベントデータ、又はジャンプイベントデータの何れかのイベントデータとなった場合は、ステップ７１６で、サーチイベントが指示する要素番号から－１して得られる値が、フラグメント終了イベントの値としてセットされる。この値が、現在処理中のフラグメント内における、トラック終了イベントデータ、小節終了イベントデータ、及びジャンプイベントデータ以外の末尾のイベントデータである。

【００５９】以上の処理により、現在処理中のフラグメント内でソート処理が実行されるべき先頭のイベントデータと末尾のイベントデータが、フラグメント先頭イベント及びフラグメント終了イベントとして算出される。

【００６０】続いて、ステップ７１７では、上記フラグメント先頭イベントとフラグメント終了イベントが指示する要素番号が一致しているか否かが判定される。ステップ７１７の判定がＹＥＳの場合は、現在処理中のフラグメント内でソート処理が実行されるべきイベントデータの数が１であり、実質的にソート処理を実行する必要はないため、ステップ７１８のソート処理は実行されずに、ステップ７１９以降が実行される。ステップ７１９以降の処理については後述する。

【００６１】ステップ７１７の判定がＮＯの場合は、ステップ７１８で、配列ＳＮＧ＿ＳＯＮＧに記憶されている対象小節に含まれる現在処理中のフラグメント内で、フラグメント先頭イベントとフラグメント終了イベントにより指示される範囲のイベントデータ群が、各イベントデータの第２ワードのクロック値に関してソートされる。この処理については、図１３の動作フローチャートを用いて後述する。なお、後述するように、ステップ７１８において実際にイベントデータの入替えが発生した場合は、ステップ７１８内において時間ソート処理終了フラグの値が０にセットされる。従って、この場合には図６のステップ６０２の処理に戻った時点でその判定がＹＥＳとなるため、時間ソート処理が続行される。

【００６２】ステップ７１８の処理の後又はステップ７１７の判定がＹＥＳとなった後に、ステップ７１９で、ステップ７１４で判定されている現在イベントデータの種類が判定される。

【００６３】ステップ７１９で現在イベントデータがト

13

ラック終了イベントデータ又は小節終了イベントデータであると判定された場合には、ステップ７２０で、フラグメント内ソート処理終了フラグの値が１にセットされる。即ち、この場合は、現在処理中のフラグメント以降に対象小節のイベントデータ群が存在しないため、ステップ７２０の処理の後にステップ７０４が実行された時点でその判定がＮＯとなり、フラグメント内ソート処理を終了する。

【００６４】また、ステップ７１９で現在イベントデータがジャンプイベントデータであると判定された場合には、ステップ７２１で、そのジャンプイベントデータの第３ワードに格納されているジャンプ先要素番号が、サーチイベントの値としてセットされ、また、サーチ状態フラグの値が０にセットされる。即ち、この場合は、現在処理中のフラグメント以降に対象小節のイベントデータ群が存在する。このため、ステップ７２１の処理によってサーチイベントに新たなフラグメントの先頭のイベントデータの要素番号が設定された後に、再びステップ７０４と７０５の判定が共にＹＥＳとなった後、ステップ７０６以降において、配列ＳＮＧ＿ＳＯＮＧに記憶されている対象小節に含まれる新たなフラグメントに対してフラグメント内ソート処理が実行される。

【００６５】以上説明した図７の動作フローチャートの処理によって、配列ＳＮＧ＿ＳＯＮＧに記憶されている対象小節内の全てのフラグメントのそれぞれに対して、図６のステップ６０４のフラグメント内ソート処理が１回ずつ実行される。

【００６６】続いて、図８～図１２は、図６のステップ６０５のフラグメント間ソート処理の動作フローチャートである。まず、フラグメント間ソート処理終了フラグ（ＣＰＵ１０３内のレジスタ又はＲＡＭ１０５に確保される）は、フラグメント間ソート処理を、終了させるべきでない場合に値０となり、終了させるべきである場合に値１となる。このフラグの値は、ステップ８０１で、０に初期化される。

【００６７】サーチイベント（ＣＰＵ１０３内のレジスタ又はＲＡＭ１０５に確保される）は、現在処理中のイベントデータを指示するものである。このサーチイベントの値は、ステップ８０２で、配列ＳＮＧ＿ＳＯＮＧに記憶される対象小節内の先頭のイベントデータの要素番号にセットされる。

【００６８】第１フラグメント最終イベント要素番号（ＣＰＵ１０３内のレジスタ又はＲＡＭ１０５に確保される）は、フラグメント間ソート処理が実行されるべき現在処理中の２つのフラグメントのうちの第１番目のフラグメントにおける、トラック終了イベントデータ、小節終了イベントデータ、ジャンプソースイベントデータ、及びジャンプイベントデータ以外の末尾のイベントデータの要素番号を指示するものである。この値は、ステップ８０３で、配列ＳＮＧ＿ＳＯＮＧ上の要素番号と

14

して出現し得ない値０ｘｆｆｆｆにセットされる。

【００６９】次に、初期状態においては、ステップ８０４の判定がＹＥＳとなった後に、ステップ８０５でＦ１フラグ（ＣＰＵ１０３内のレジスタ又はＲＡＭ１０５に確保される）の値が０にセットされる。Ｆ１フラグは、配列ＳＮＧ＿ＳＯＮＧに記憶されている対象小節に含まれる現在処理中の２つのフラグメントのうちの第１番目のフラグメントにおいて、トラック終了イベントデータ、小節終了イベントデータ、ジャンプソースイベントデータ、及びジャンプイベントデータ以外の末尾のイベントデータをサーチする処理が、終了していない場合に値０となり、終了している場合に値１となる。

【００７０】続いて、図９のステップ８０６の判定がＹＥＳとなった後、図９のステップ８０７～８１４で、配列ＳＮＧ＿ＳＯＮＧに記憶されている対象小節に含まれる現在処理中の２つのフラグメントのうちの第１番目のフラグメントにおいて、トラック終了イベントデータ、小節終了イベントデータ、ジャンプソースイベントデータ、及びジャンプイベントデータ以外の末尾のイベントデータをサーチする処理が実行される。

【００７１】まず、ステップ８０７で、サーチイベントにより指示される現在イベントデータが、ＲＡＭ１０５内の配列ＳＮＧ＿ＳＯＮＧから読み出される。そして、そのイベントデータの種類が判定される。

【００７２】ステップ８０７で現在イベントデータがトラック終了イベントデータ又は小節終了イベントデータであると判定された場合には、ステップ８０８で、Ｆ１フラグとフラグメント間ソート処理終了フラグの値が共に１にセットされる。即ち、この場合は、現在処理中の第１番目のフラグメント内にはソート処理を実行すべき有効な最終イベントデータは存在せず、かつ第１番目のフラグメント以降に対象小節のイベントデータ群が存在しないため、ステップ８０８の処理の後にステップ８０６が実行された時点でその判定がＮＯとなり、後述する図１０のステップ８１５が実行された後、ステップ８１６に対応する後述する図１１の動作フローチャートが実行される時点でステップ１１０１の判定がＮＯとなり、更に、図１０のステップ８１７の判定がＮＯ、図８のステップ８０４の判定がＮＯとなって、フラグメント間ソート処理を終了する。

【００７３】また、図９のステップ８０７で現在イベントデータがジャンプイベントデータであると判定された場合には、ステップ８０９で、第１フラグメント最終イベント要素番号の値がステップ８０３で初期設定されたままの値０ｘｆｆｆｆであるか否かが判定される。

【００７４】今、配列ＳＮＧ＿ＳＯＮＧ内の現在処理中の２つのフラグメントのうちの第１番目のフラグメント内に、トラック終了イベントデータ、小節終了イベントデータ、ジャンプソースイベントデータ、及びジャンプイベントデータ以外のイベントデータが１つも存在して

15

いなければ、第１フラグメント最終イベント要素番号の値は初期設定値０ｘｆｆｆｆのままとなり、ステップ８０９の判定がＹＥＳとなる。この場合には、ステップ８１１で、そのジャンプイベントデータの第３ワードに格納されている新たなフラグメントを示すジャンプ先要素番号が、サーチイベントの値としてセットされる。その後、再びステップ８０６の判定がＹＥＳとなることにより、ステップ８０７～８１４において、配列ＳＮＧ＿ＳＯＮＧに記憶されている対象小節に含まれる新たなフラグメントが現在処理中の２つのフラグメントのうちの第１番目のフラグメントとされて、処理が続行される。

【００７５】更に、ステップ８０７で現在イベントデータがトラック終了イベントデータ、小節終了イベントデータ、及びジャンプイベントデータ以外のイベントデータであると判定された場合において、更にステップ８１２で現在イベントデータがジャンプソースイベントデータであると判定された場合には、ステップ８１４で、単純にサーチイベントが指示する配列ＳＮＧ＿ＳＯＮＧの要素番号がインクリメントされた後に、再びステップ８０６からステップ８０７の処理に戻る。

【００７６】また、ステップ８０７で現在イベントデータがトラック終了イベントデータ、小節終了イベントデータ、及びジャンプイベントデータ以外のイベントデータであると判定された場合において、更にステップ８１２で現在イベントデータがジャンプソースイベントデータ以外の通常のイベントデータであると判定された場合には、ステップ８１３で、サーチイベントの値が第１フラグメント最終イベント要素番号としてセットされる。そして、ステップ８１４で、サーチイベントが指示する配列ＳＮＧ＿ＳＯＮＧの要素番号がインクリメントされた後に、再びステップ８０６からステップ８０７の処理に戻る。

【００７７】以上のようにして第１フラグメント最終イベント要素番号がインクリメントされてゆき、最後にステップ８０７でジャンプイベントデータが検出された時点において第１フラグメント最終イベント要素番号にセットされている要素番号が、配列ＳＮＧ＿ＳＯＮＧに記憶されている対象小節に含まれる現在処理中の２つのフラグメントのうちの第１番目のフラグメントにおいて見つかった、トラック終了イベントデータ、小節終了イベントデータ、ジャンプソースイベントデータ、及びジャンプイベントデータ以外の末尾のイベントデータの要素番号となる。その後、ステップ８０９の判定がＮＯとなって、第１番目のフラグメントにおけるサーチ処理を終了させるべく、ステップ８１０で、Ｆ１フラグの値が１にセットされる。そして、ステップ８１１で、そのジャンプイベントデータの第３ワードに格納されている新たなフラグメントを示すジャンプ先要素番号が、サーチイベントの値としてセットされる。

【００７８】この結果、ステップ８１１に続いて実行されるステップ８０６の判定がＮＯとなり、図１０のステップ８１５でＦ２フラグ（ＣＰＵ１０３内のレジスタ又はＲＡＭ１０５に確保される）の値が０にセットされる。Ｆ２フラグは、配列ＳＮＧ＿ＳＯＮＧに記憶されている対象小節に含まれる現在処理中の２つのフラグメントのうちの第２番目のフラグメントにおいて、トラック終了イベントデータ、小節終了イベントデータ、ジャンプソースイベントデータ、及びジャンプイベントデータ以外の先頭のイベントデータをサーチする処理が、終了していない場合に値０となり、終了している場合に値１となる。

16

【００７９】続いて、図１０のステップ８１６で、配列ＳＮＧ＿ＳＯＮＧに記憶されている対象小節に含まれる現在処理中の２つのフラグメントのうちの第２番目のフラグメントにおいて、トラック終了イベントデータ、小節終了イベントデータ、ジャンプソースイベントデータ、及びジャンプイベントデータ以外の先頭のイベントデータをサーチする処理が実行される。

【００８０】図１１は、図１０のステップ８１６の処理の詳細を示す動作フローチャートである。まず、最初はステップ１１０１の判定がＹＥＳとなる。

【００８１】次に、ステップ１１０２で、サーチイベントにより指示される現在イベントデータが、ＲＡＭ１０５内の配列ＳＮＧ＿ＳＯＮＧから読み出される。そして、そのイベントデータの種類が判定される。

【００８２】ステップ１１０２で現在イベントデータがトラック終了イベントデータ又は小節終了イベントデータであると判定された場合には、ステップ１１０３で、フラグメント間ソート処理終了フラグの値が１にセットされる。即ち、この場合は、現在処理中の２つのフラグメントのうちの第２番目のフラグメント内にはソート処理を実行すべき有効な先頭イベントデータは存在せず、かつ第２番目のフラグメント以降に対象小節のイベントデータ群が存在しないため、ステップ１１０３の処理の後にステップ１１０１が実行された時点でその判定がＮＯとなり、図１１の動作フローチャートによって示される図１０のステップ８１６の処理を終了する。

【００８３】また、ステップ１１０２で現在イベントデータがジャンプイベントデータであると判定された場合には、ステップ１１０４で、そのジャンプイベントデータの第３ワードに格納されているジャンプ先要素番号が、サーチイベントの値としてセットされる。即ち、この場合には、現在処理中の２つのフラグメントのうちの第２番目のフラグメント内にはソート処理を実行すべき有効なイベントデータは存在せず、かつその第２番目のフラグメント以降に対象小節のイベントデータ群が存在する。このため、ステップ１１０４の処理によってサーチイベントに新たなフラグメントの先頭のイベントデータの要素番号が設定される。その後、再びステップ１１０１の判定がＹＥＳとなることにより、ステップ１１０

２～１１０８において、配列ＳＮＧ＿ＳＯＮＧに記憶されている対象小節に含まれる新たなフラグメントが現在処理中の２つのフラグメントのうちの第２番目のフラグメントとされて、処理が続行される。

【００８４】更に、ステップ１１０２において現在イベントデータがトラック終了イベントデータ、小節終了イベントデータ、及びジャンプイベントデータ以外のイベントデータであると判定された場合において、更にステップ１１０５で現在イベントデータがジャンプソースイベントデータであると判定された場合には、ステップ１１０８で、単純にサーチイベントが指示する配列ＳＮＧ＿ＳＯＮＧの要素番号がインクリメントされた後に、再びステップ１１０１からステップ１１０２の処理に戻る。

【００８５】また、ステップ１１０２で現在イベントデータがトラック終了イベントデータ、小節終了イベントデータ、及びジャンプイベントデータ以外のイベントデータであると判定された場合において、更にステップ１１０５で現在イベントデータがジャンプソースイベントデータ以外の通常のイベントデータであると判定された場合には、ステップ１１０６で、サーチイベントの値が第２フラグメント先頭イベント要素番号としてセットされる。この値が、配列ＳＮＧ＿ＳＯＮＧに記憶されている対象小節に含まれる現在処理中の２つのフラグメントのうちの第２番目のフラグメントにおいて最初に見つかった、トラック終了イベントデータ、小節終了イベントデータ、ジャンプソースイベントデータ、及びジャンプイベントデータ以外の先頭のイベントデータの要素番号となる。続いて、第２番目のフラグメントにおけるサーチ処理を終了させるべく、ステップ１１０７で、Ｆ２フラグの値が１にセットされた後、ステップ１１０８で、サーチイベントが指示する配列ＳＮＧ＿ＳＯＮＧの要素番号がインクリメントされる。そして、ステップ１１０６～１１０８が実行された後にステップ１１０１が再び実行される時点で、その判定がＮＯとなり、図１１によって示される図１０のステップ８１６の処理を終了する。

【００８６】以上のようにして、配列ＳＮＧ＿ＳＯＮＧに記憶されている対象小節に含まれる現在処理中の２つのフラグメントにつき、図９のステップ８０７～８１４で、第１番目のフラグメントの末尾のイベントデータをサーチする処理が成功し、かつ、図１１によって示される図１０のステップ８１６で、第２番目のフラグメントの先頭のイベントデータをサーチする処理に成功すると、ステップ８１７の判定がＹＥＳとなった後、ステップ８１８で、第１番目のフラグメントの末尾のイベントデータと第２番目のフラグメントの先頭のイベントデータについて、イベント移動処理が実行される。

【００８７】図１２は、図１０のステップ８１８のイベント移動処理の動作フローチャートである。まず、ステップ１２０１で、ＲＡＭ１０５内の配列ＳＮＧ＿ＳＯＮＧから、第１フラグメント最終イベント要素番号によって指示される要素番号のイベントデータが読み出されると共に、第２フラグメント先頭イベント要素番号によって指示される要素番号のイベントデータが読み出される。そして、第１フラグメント最終イベント要素番号に対応するイベントデータの第２ワードの値が、第２フラグメント先頭イベント要素番号に対応するイベントデータの第２ワードの値より大きいか否か、即ち、第１フラグメント最終イベント要素番号に対応するイベントデータが第２フラグメント先頭イベント要素番号に対応するイベントデータよりも後ろに配置されるべきか否かが判定される。

【００８８】ステップ１２０１の判定がＹＥＳの場合には、ステップ１２０２で、第１フラグメント最終イベント要素番号に対応するイベントデータと第２フラグメント先頭イベント要素番号に対応するイベントデータの記憶位置が入れ替えられる。その後、ステップ１２０３及び１２０４で、フラグメント間ソート処理終了フラグ及び時間ソート処理終了フラグの値が、共に０にセットされる。このように、実際にイベントデータの入替えが発生した場合には、時間ソート処理終了フラグの値が０にセットされるため、前述した図６のステップ６０４のフラグメント内ソート処理とステップ６０５のフラグメント間ソート処理が終了した後にステップ６０２の判定に戻った時点で、その判定が再びＹＥＳとなり、時間ソート処理が続行される。

【００８９】一方、ステップ１２０１の判定がＮＯの場合には、第１フラグメント最終イベント要素番号に対応するイベントデータと第２フラグメント先頭イベント要素番号に対応するイベントデータの入替えは実行されず、ステップ１２０５で、フラグメント間ソート処理終了フラグの値が０にセットされる。

【００９０】以上のようにして、配列ＳＮＧ＿ＳＯＮＧに記憶される対象小節に含まれる現在処理中の２つのフラグメントについて、フラグメント間ソート処理が実現される。

【００９１】ステップ１２０４又は１２０５の処理の後、再び図８のステップ８０４の処理に戻り、その判定がＹＥＳとなることにより、ステップ８０５以降で、配列ＳＮＧ＿ＳＯＮＧに記憶されている対象小節に含まれる新たな２つのフラグメントにつき、第１番目のフラグメントの末尾のイベントデータをサーチする処理と、第２番目のフラグメントの先頭のイベントデータをサーチする処理と、サーチされた２つのイベントデータに対するイベント移動処理の実行が試みられる。この場合、サーチイベントの値は今まで第２番目であったフラグメントの先頭のイベントデータを指示しているため、今まで第２番目として処理されたフラグメントが新たな第１番目のフラグメントとして処理される。そして、それに続

く対象小節内のフラグメントが第２番目のフラグメントとして処理される。

【００９２】以上の処理が繰返される結果、対象小節の末尾のフラグメントが第１番目のフラグメントとして処理される過程において、ステップ８０７でトラック終了イベントデータ又は小節終了イベントデータが検出される。これにより、ステップ８０８で、Ｆ１フラグとフラグメント間ソート処理終了フラグの値が共に１にセットされる。即ち、この場合は、現在処理中の第１番目のフラグメント内にはソート処理を実行すべき有効な最終イベントデータは存在せず、かつ第１番目のフラグメント以降に対象小節のイベントデータ群が存在しないため、ステップ８０８の処理の後にステップ８０６が実行された時点でその判定がＮＯとなり、後述する図１０のステップ８１５が実行された後、ステップ８１６に対応する後述する図１１の動作フローチャートが実行される時点でステップ１１０１の判定がＮＯとなり、更に、図１０のステップ８１７の判定がＮＯ、図８のステップ８０４の判定がＮＯとなって、フラグメント間ソート処理を終了する。

【００９３】最後に、図１３は、図７のステップ７１８において実行される、配列ＳＮＧ＿ＳＯＮＧに記憶されている対象小節に含まれる現在処理中のフラグメント内の、フラグメント先頭イベントとフラグメント終了イベントにより指示される範囲のイベントデータ群に対する時間ソート処理を示す動作フローチャートである。

【００９４】ここでは、フラグメント先頭イベントとフラグメント終了イベントにより指示される範囲のイベントデータ群が、各イベントデータの第２ワードのクロック値に関してソートされる。

【００９５】以下の説明では、図１３の動作フローチャートについて説明する前に、ソートアルゴリズムの代表例と、本実施例で採用する改良アルゴリズムの原理について簡単に説明する。なお、以下で説明するバブルソートアルゴリズムとコムソートアルゴリズムの詳細な原理については、文献「バブル・ソートが簡単な手直しで劇的に速くなる」（スティーブン・レイシー、リチャード・ボックス）（日経ＢＰ社発行、日経バイト／１９９１年１１月号、ｐ．３０５～ｐ．３１２）に記載されている。

【００９６】ソートアルゴリズムの最も代表的なものとして、バブルソートアルゴリズムが知られている。このアルゴリズムにおいては、例えば昇べきの順にソートが行われる場合には、まず、第１番目のイベントデータ値（第２バイトの値）と第２番目のイベントデータ値の大小関係が比較され、第１番目のイベントデータ値の方が大きければ両者のイベントデータが入れ替えられる。次に、新たな第２番目のイベントデータ値と第３番目のイベントデータ値の大小関係が同様に比較され入替えが実行される。このようにして、最終イベントデータまで同

様の比較・入替え処理が実行される。この最終データまでの１回の処理単位を、ストロークと呼ぶ。続いて、再び、第１番目のイベントデータから最終イベントデータに向かって第２ストローク目の処理が繰り返される。そして、このストローク処理が、イベントデータの入替えが発生しなくなるまで繰り返される。

【００９７】以上のバブルソートアルゴリズムは、隣り同士のイベントデータ値の比較・入替え処理の繰返しによって実現されるため、処理プログラムが簡単に実現できる反面、イベントデータは１回の比較・入替え処理によって１アドレス分だけしか移動できないため、例えばフラグメントの後部に位置するイベントデータがフラグメントの先頭付近に移動させられる場合には処理時間が長くなってしまう。

【００９８】そこで、そのようなバブルソートアルゴリズムの欠点を改良するアルゴリズムとして、コムソートアルゴリズムが知られている。このアルゴリズムでは、バブルソートアルゴリズムのように隣接したイベントデータ値同士について比較・入替えが実行されるのではなく、１以上離れた間隔（以下、この間隔をギャップという）のイベントデータ値同士について、比較・入替えが実行される。

【００９９】具体的には、次のようなコムソートアルゴリズムが最適であることが実験的に確かめられている。即ち、まず、最初のストローク処理（第１番目のイベントデータから最終イベントデータまでの１回の処理）におけるギャップの値は、ソート処理されるイベントデータの要素数を１．３で割って得られる値とされる。以下、ストロークが進む毎に、前回のギャップの値を１．３で割って得られる値が新たなギャップの値とされる。この過程で、ギャップの値が９又は１０となった時点で、ギャップの値が強制的に１１に置き換えられて処理が続行される。商が１未満になると、それ以降のギャップの値は１とされる。そして、ギャップの値が１となり、かつストローク処理においてイベントデータの入替えが発生しなくなった時点で、ソート処理が終了する。

【０１００】以上のコムソートアルゴリズムが、最も高速なソートアルゴリズムとして知られている。しかし、上述のコムソートアルゴリズムでは、ストロークが進む毎にソート処理されるべき要素数又は前回のギャップ値を１．３で割ることにより新たなギャップの値を算出する処理が必要となる。このことは、上記除算処理のために小数演算用コプロセッサが必要になるということを意味するため、ハードウエア規模が大きくなってしまうという欠点がある。

【０１０１】そこで、本実施例では、コムソートアルゴリズムを基本とし、値１．３による除算処理を、値３の乗算処理と値４の除算処理で置き換えたアルゴリズムが採用される。即ち、値１．３による除算処理が値（３／４）による除算処理に置き換えられたことになる。これ

により、小数演算用コプロセッサが不要となり、ハードウエア規模の増大を抑えることができる。なお、このようにギャップ値が代用されても、ソート時間の増大はそれほど大きくならない。

【０１０２】図１３は、上述のアルゴリズムに基づく図７のステージ７１８の処理の詳細な動作フローチャートである。図１３の動作フローチャートにおいて、まず、ＳＩＺＥ（ＣＰＵ１０３内のレジスタ又はＲＡＭ１０５に確保される）には、ソート処理の対象となるフラグメントの要素数がセットされる。この数は、（フラグメント終了イベント－フラグメント先頭イベント＋１）として算出できる。

【０１０３】次に、ＧＡＰ（ＣＰＵ１０３内のレジスタ又はＲＡＭ１０５に確保される）には、前述のギャップの値がセットされる。また、ＴＯＰ（ＣＰＵ１０３内のレジスタ又はＲＡＭ１０５に確保される）には、後述するイベント移動ループ処理時の上限のイベントの要素値がセットされる。

【０１０４】更に、ＣＯＵＮＴＥＲ（ＣＰＵ１０３内のレジスタ又はＲＡＭ１０５に確保される）には、後述する移動イベントカウンタ値がセットされる。図１３の動作フローチャートにおいて、まず、ステップ１３０１で、ＧＡＰにＳＩＺＥの値がセットされる。次に、前述した除算処理の簡略化原理に基づき、ステップ１３０２において、ＧＡＰの値に整数値３を乗算する整数乗算処理が実行され、更に、その乗算結果を整数値４で除算する整数除算処理が実行される。この除算処理は、実際には、２ビット右シフト演算として実現できる。この演算結果が新たなＧＡＰの値としてセットされる。

【０１０５】次に、ステップ１３０３で新たに算出されたＧＡＰの値が判定され、ＧＡＰの値が９又は１０の場合には、前述したコムソート１１のアルゴリズムの原理に基づいて、ステップ１３０４で、ＧＡＰの値が１１に修正される。また、ＧＡＰの値が０となってしまった場合には、やはり前述したコムソート１１のアルゴリズムの原理に基づいて、ステップ１３０５で、ＧＡＰの値が１に修正される。ＧＡＰの値が９、１０、又は０の何れでもない場合には、ＧＡＰの値の修正は行われない。

【０１０６】次に、ステップ１３０６で、ＣＯＵＮＴＥＲの値が０にセットされる。続いて、ステップ１３０７で、第１イベント要素番号（ＣＰＵ１０３内のレジスタ又はＲＡＭ１０５に確保される）に、図７の動作フローチャートにおいて求まっているフラグメント先頭イベントにセットされている要素番号がセットされる。この第１イベント要素番号は、後述するイベント移動ループ処理において比較される２つのイベントデータのうち要素番号が小さい方のイベントデータに対応する。

【０１０７】次に、ＴＯＰの値が、次の数１式により算出されセットされる。

【０１０８】

【数１】

ＴＯＰ＝フラグメント先頭イベント＋ＳＩＺＥ－ＧＡＰ

このＴＯＰの値は、次に説明するイベント移動ループ処理においてインクリメントすることのできる第１イベント要素番号の上限値に対応する。

【０１０９】続いて、ステップ１３０９～１３１４→１３１５→１３０９のイベント移動ループ処理が実行される。ここでは、１ストローク分のイベントデータの比較・入替え処理が実行される。

【０１１０】まず、ステップ１３０９では、第２イベント要素番号（ＣＰＵ１０３内のレジスタ又はＲＡＭ１０５に確保される）の値が、次の数２式によって算出されセットされる。

【０１１１】

【数２】

第２イベント要素番号＝第１イベント要素番号＋ＧＡＰ

この第２イベント要素番号は、イベント移動ループ処理において比較される２つのイベントデータのうち要素番号が大きい方のイベントデータに対応する。

【０１１２】ステップ１３１０では、ＲＡＭ１０５の配列ＳＮＧ＿ＳＯＮＧ（図３(c)）に記憶される第１イベント要素番号に対応するイベントデータの第２ワードの小節先頭からのクロック値が、同じく第２イベント要素番号に対応するイベントデータの第２ワードの小節先頭からのクロック値よりも大きいか否か、即ち、第１イベント要素番号に対応するイベントデータが第２イベント要素番号に対応するイベントデータよりも後ろに配置されるべきか否かが判定される。

【０１１３】ステップ１３１０の判定がＹＥＳの場合には、ステップ１３１１で、第１イベント要素番号に対応するイベントデータと第２イベント要素番号に対応するイベントデータの記憶位置が入れ替えられる。その後、ステップ１３１２で、ＣＯＵＮＴＥＲの値がインクリメントされる。更に、ステップ１３１３で、時間ソート処理終了フラグの値が、０にセットされる。このように、実際にイベントデータの入替えが発生した場合には、時間ソート処理終了フラグの値が０にセットされるため、前述した図６のステップ６０４のフラグメント内ソート処理とステップ６０５のフラグメント間ソート処理が終了した後にステップ６０２の判定に戻った時点で、その判定が再びＹＥＳとなり、時間ソート処理が続行される。

【０１１４】一方、ステップ１３１０の判定がＮＯの場合には、ステップ１３１１～１３１３の処理は実行されない。続いて、ステップ１３１４で、第１イベント要素番号の値がＴＯＰの値より小さいか否かが判定される。

【０１１５】ステップ１３１４の判定がＹＥＳなら、ステップ１３１５で第１イベント要素番号の値がインクリメントされた後、ステップ１３０９で第１イベント要素番号からＧＡＰの値分だけ離れた新たな第２イベント要

素番号の値が算出され、ステップ１３１０～１３１３で、両者に対する比較・入替えの処理が実行される。

【０１１６】以上のステップ１３０９～１３１５の処理がステップ１３１４の判定がＮＯとなるまで繰り返されることにより、１ストローク分のイベントデータの比較・入替え処理が実行される。

【０１１７】次に、ステップ１３１４の判定がＮＯとなり１ストローク分のイベントデータの比較・入替え処理が終了すると、ステップ１３１５で、ＣＯＵＮＴＥＲの値が０でないか（即ち、直前にイベントデータの入替えが行われたか）、又はＧＡＰの値が１より大きいかが判定される。

【０１１８】ステップ１３１６の判定がＹＥＳなら、ステップ１３０２～１３０５に戻って新たなストロークのためのＧＡＰの値が算出され、ステップ１３０６～１３０８の処理の後、前述したステップ１３０９～１３１５のイベント移動ループ処理によって、新たなストロークに対するイベントデータの比較・入替え処理が繰り返される。

【０１１９】ＧＡＰの値が１となり、かつ前述したイベント移動ループ処理によってイベントデータの入替えが発生しなくなり、ステップ１３１６の判定がＮＯとなった時点で、図１３によって示される図７のステップ７１８のフラグメント内のソート処理が終了する。

【０１２０】

【発明の効果】本発明によれば、例えば楽曲データの再生処理において、非常に高速な１又は複数小節単位の早送り再生又は巻き戻し再生を簡単な制御で実現することが可能となる。

【０１２１】また、楽曲データの編集処理が実行される場合に、例えば小節単位の移動、既存の小節への新たな楽曲データの追加、その他、楽曲データの挿入、削除等を、簡単なデータ操作で実現することが可能となる。

【０１２２】このように、楽曲データが、第１の記憶手段、第２の記憶手段、又は第３の記憶手段を介して階層的に管理されることにより、楽曲データの再生処理、編集処理等の制御を簡単に実行することが可能となる。

【図面の簡単な説明】

【図１】本発明の実施例の構成図である。

【図２】ＲＡＭ１０５に記憶される楽曲データのデータフォーマットを示す図（その１）である。

【図３】ＲＡＭ１０５に記憶される楽曲データのデータフォーマットを示す図（その２）である。

【図４】ノートイベントデータとコントロールイベントデータのデータフォーマットを示す図である。

【図５】ジャンプイベントデータ群のデータフォーマットを示す図である。

【図６】時間ソート処理の全体動作フローチャートである。

【図７】フラグメント内ソート処理の動作フローチャートである。

【図８】フラグメント間ソート処理の動作フローチャート（その１）である。

【図９】フラグメント間ソート処理の動作フローチャート（その２）である。

【図１０】フラグメント間ソート処理の動作フローチャート（その３）である。

【図１１】第２フラグメント先頭イベントサーチ処理の動作フローチャートである。

【図１２】イベント移動処理の動作フローチャートである。

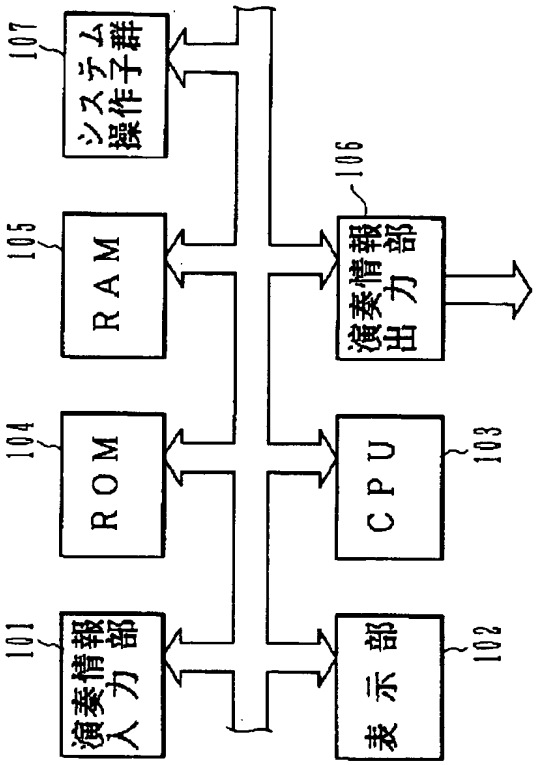
【図１３】フラグメント内のイベントソート処理を示す動作フローチャートである。

【符号の説明】

１０１　演奏情報入力部
１０２　表示部
１０３　ＣＰＵ
１０４　ＲＯＭ
１０５　ＲＡＭ
１０６　演奏情報出力部
１０７　システム操作子群

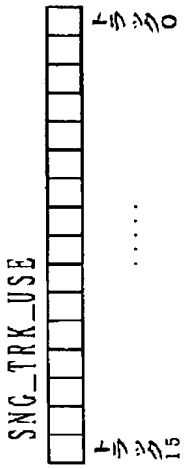
【図1】

本発明の実施例の構成図



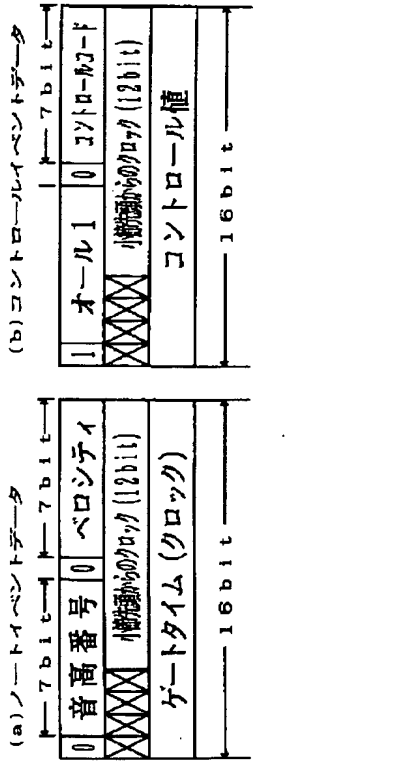
【図2】

RAM105に記憶される楽曲データのデータフォーマットを示す図（その1）



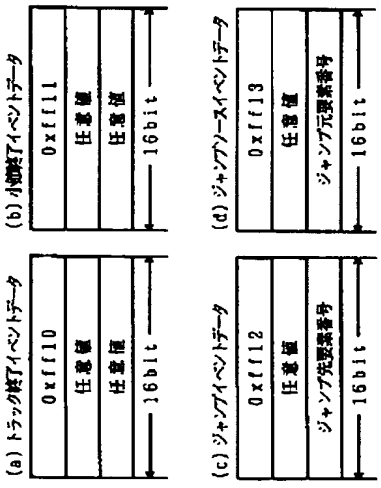
【図4】

ノートイベントデータとコントロールイベントデータのデータフォーマットを示す図



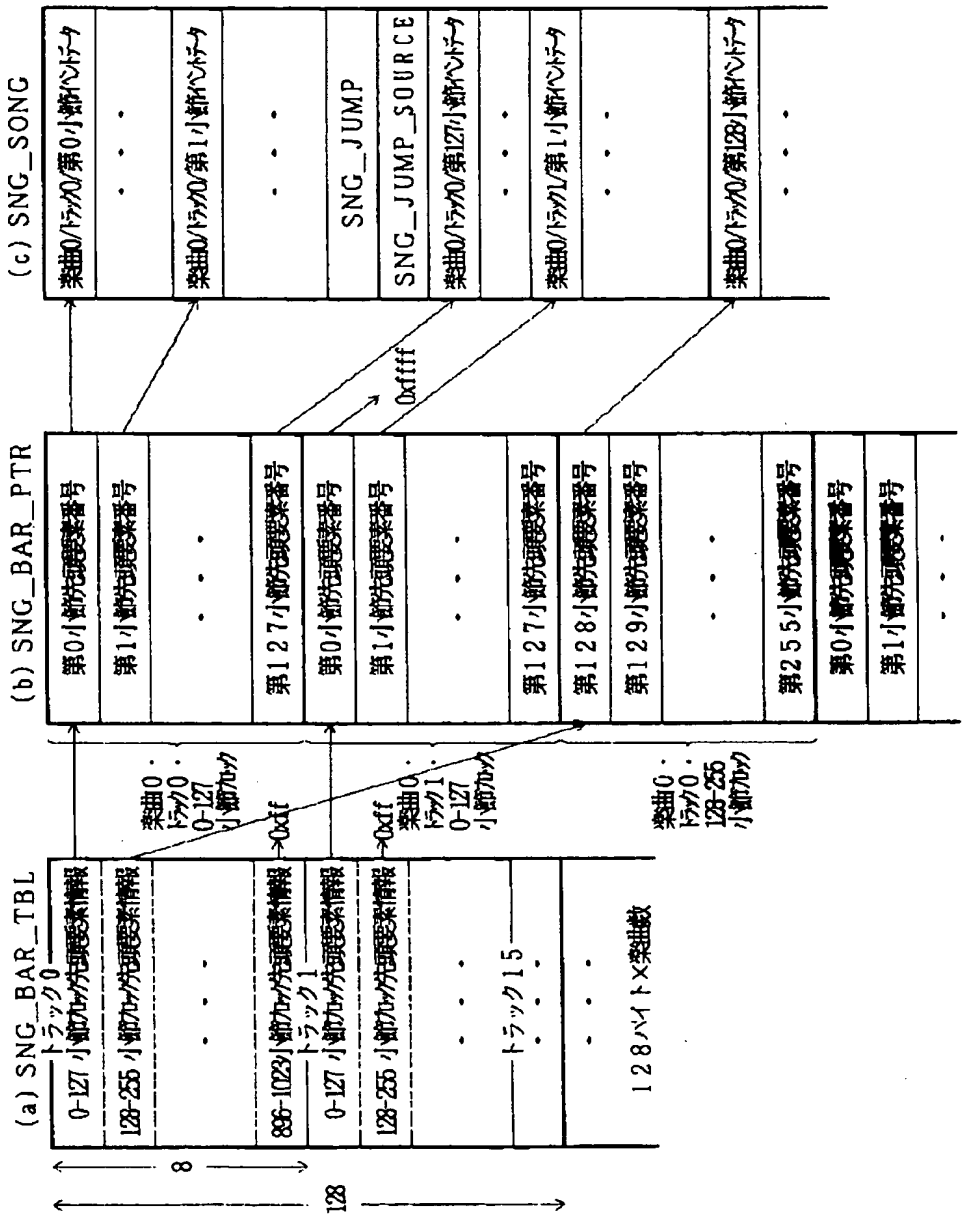
【図5】

ジャンプイベントデータ群のデータフォーマットを示す図



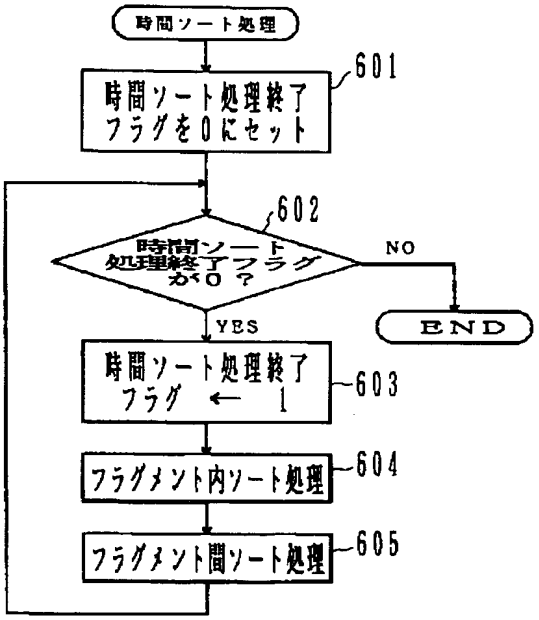
【図３】

RAM105に記憶される楽曲データの
データフォーマットを示す図（その2）



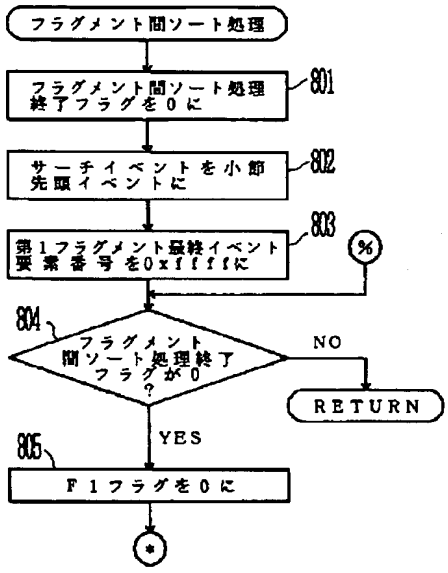
【図６】

時間ソート処理の全体動作フローチャート



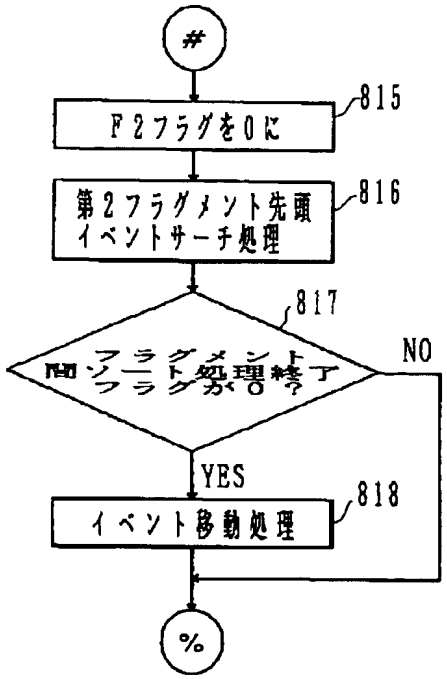
【図８】

フラグメント間ソート処理の動作フローチャート（その1）



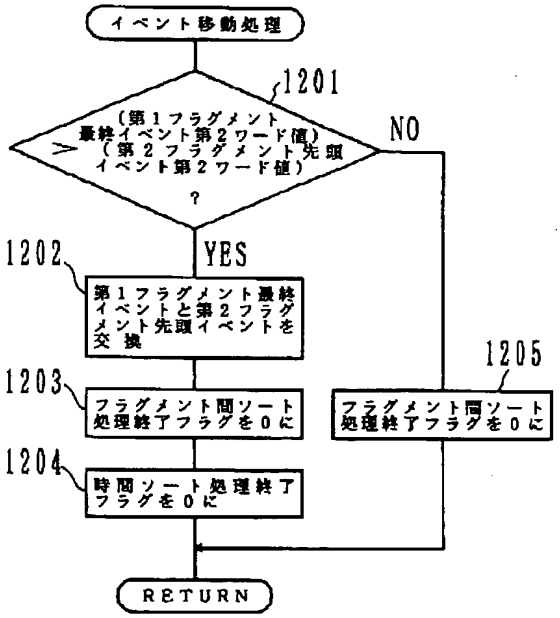
【図１０】

フラグメント間ソート処理の動作フローチャート（その3）



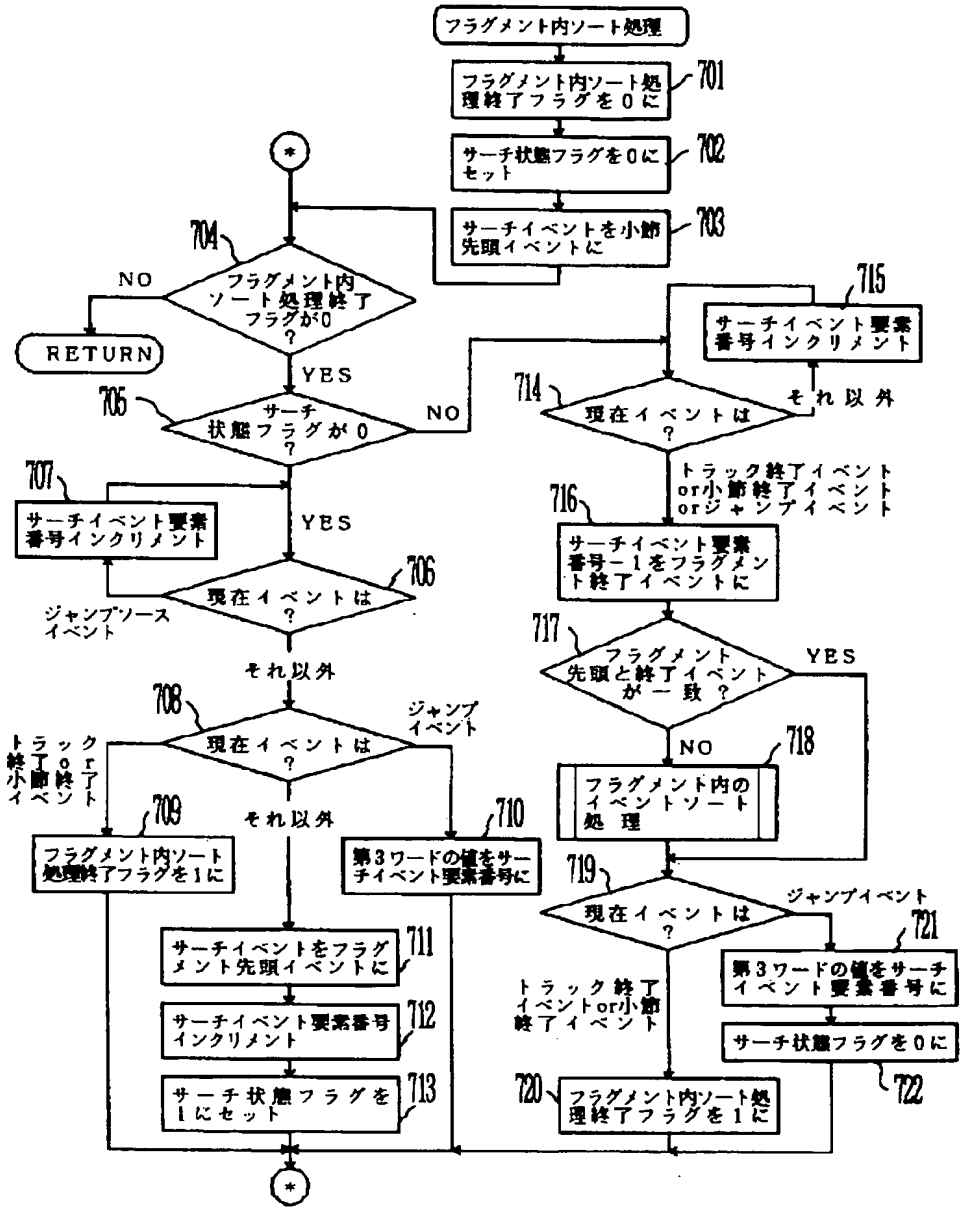
【図１２】

イベント移動処理の動作フローチャート



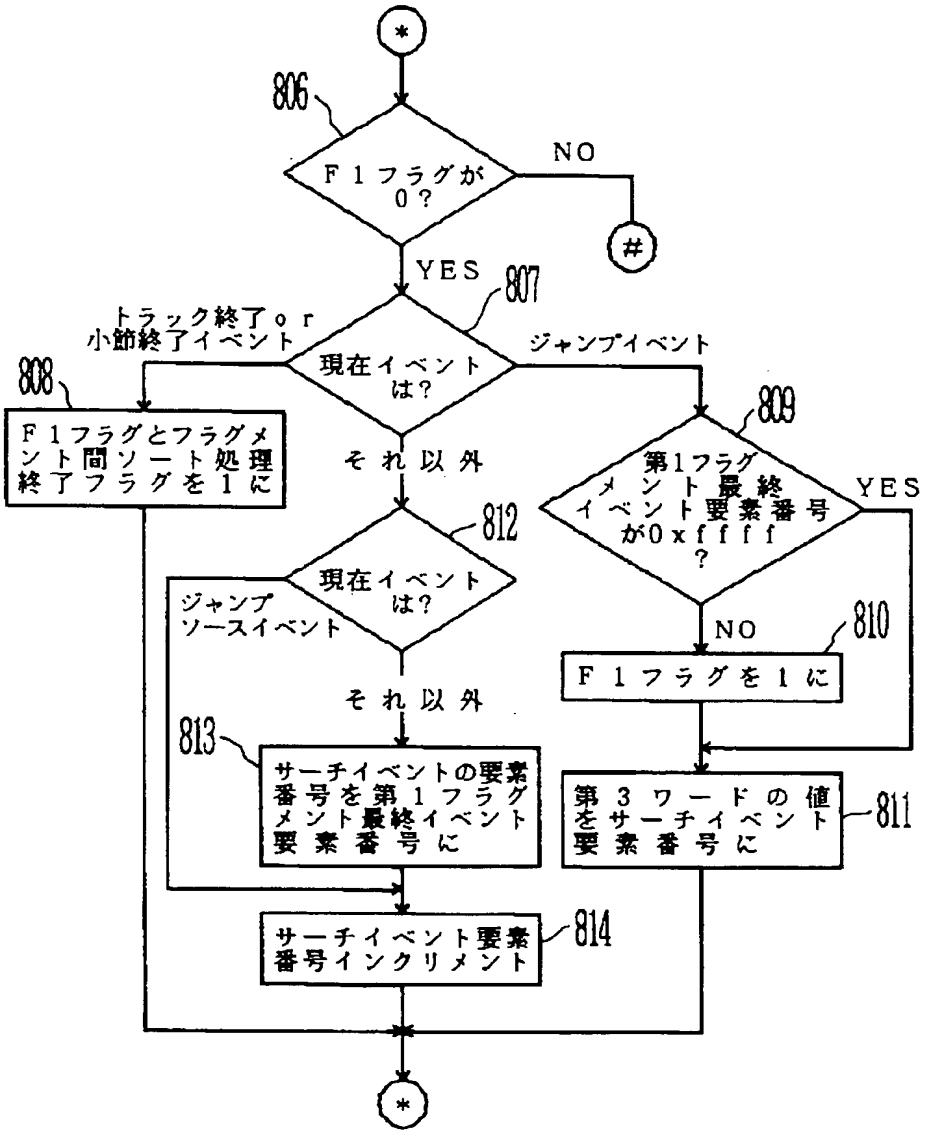
【図7】

フラグメント内ソート処理の動作フローチャート



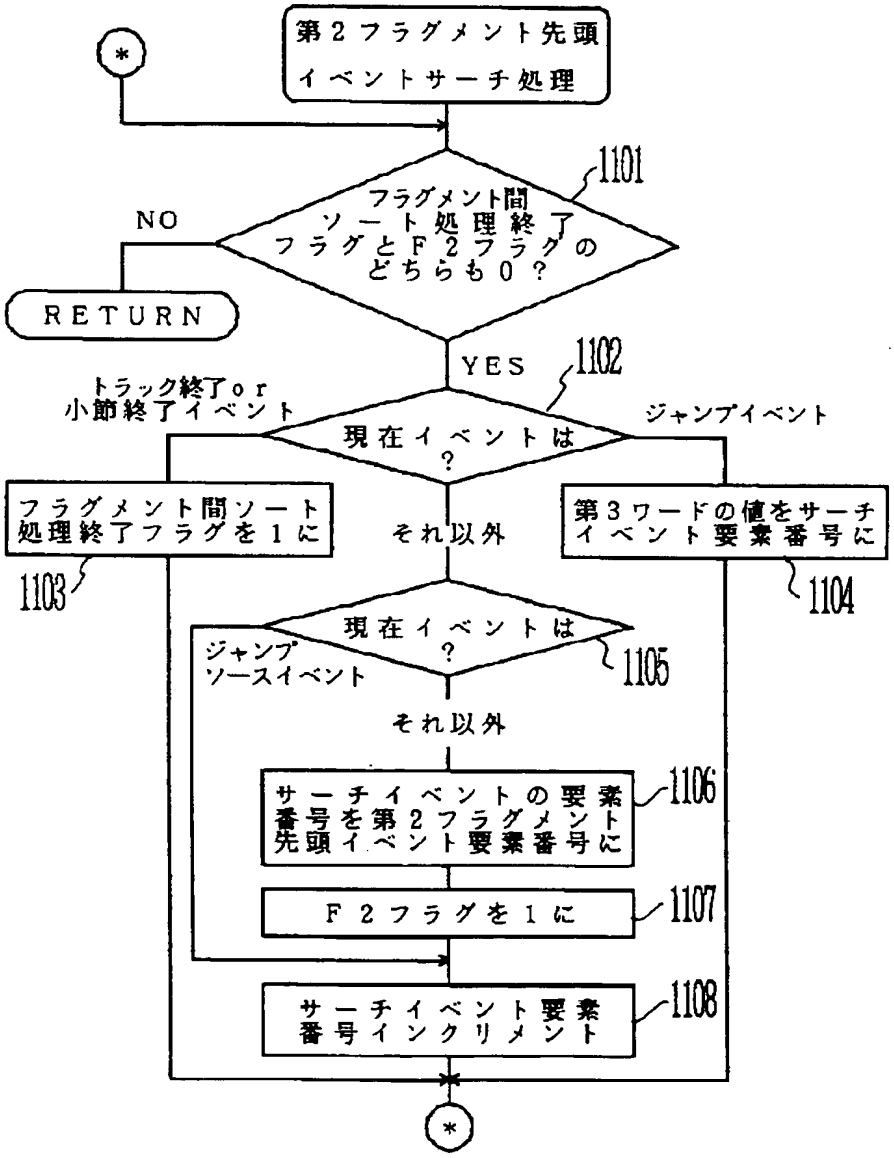
【図９】

フラグメント間ソート処理の動作フローチャート（その２）



【図１１】

第２フラグメント先頭イベントサーチ処理の動作フローチャート



【図１３】

フラグメント内のイベントソート処理を示す動作フローチャート